GOBIERNO DEL
ESTADO DE MÉXICO

TESCHA
TECNOLÓGICO DE ESTUDIOS SUPERIORES
CHALCO

TESCHA

Practica 1

Materia: Lenguajes de Interfaz

Ing. Ladislao Roberto Aldama Rojano

Integrantes:
Alvarado Guarneros Bryan
Hernández Lazcano Tomas
Romero Méndez Anel Azari
Villagómez Flores Adrián

Grupo: 4601

Turno: Matutino

MARCO TEÓRICO

MICROCONTROLADOR

Un microcontrolador (µC, UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.

Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Algunos microcontroladores pueden utilizar palabras de cuatro bits y funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia (mW o microwatts). Por lo general, tendrá la capacidad de mantenerse a la espera de un evento como pulsar un botón o de otra interrupción; así, el consumo de energía durante el estado de reposo (reloj de la CPU y los periféricos de la mayoría) puede ser sólo de nanowatts, lo que hace que muchos de ellos sean muy adecuados para aplicaciones con batería de larga duración. Otros microcontroladores pueden servir para roles de rendimiento crítico, donde sea necesario actuar más como un procesador digital de señal (DSP), con velocidades de reloj y consumo de energía más altos.

MPLAB 8.33

Es un editor IDE gratuito, destinado a productos de la marca Microchip. Este editor es modular, permite seleccionar los distintos microcontroladores soportados, además de permitir la grabación de estos circuitos integrados directamente al programador. Es un programa que corre bajo Windows, Mac OS y Linux.

PROTEUS 8.1

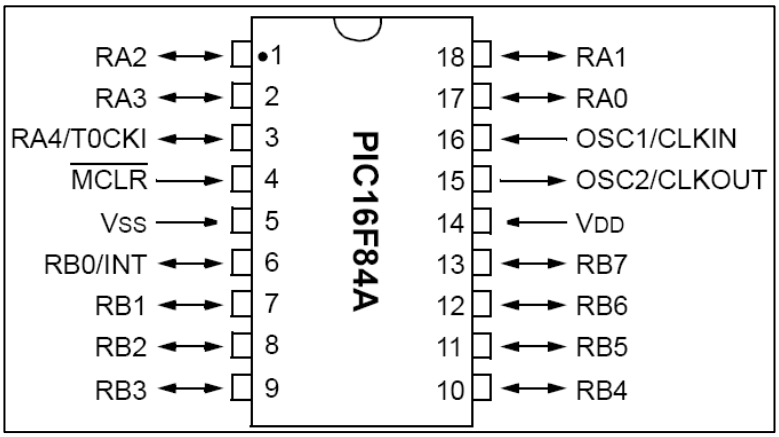
Proteus es una aplicación para la ejecución de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño del esquema electrónico, programación del software, construcción de la placa de circuito impreso, simulación de todo el conjunto, depuración de errores, documentación y construcción

PIC16F84A

El PIC16F84A tiene 68 bytes de RAM y 64 bytes de EEPROM de datos, tiene 12 registros de función específica FSR. Con 8 niveles de Pila hardware y tres modos de direccionamiento: directo, indirecto y relativo. Dispone de cuatro fuentes de interrupción y 13 pines de E/S (puertoA de 5 bits y el puertoB de 8 bits), con control individual bidireccional y dispone de un Timer0/ Contador con reloj independiente y la gran ventaja dispone de Perro Guardián (WDT). Este dispositivo tiene otras particularidades que lo hacen seriamente aconsejable, algunas de estas se detallan en otros artículos de esta serie y en las hojas de características del fabricante Microchip.

El PIC16F84A tiene un contador de programa de 13 bits capaz de dirigir 8 kilobytes x 14 espacio de memoria de programa. Para el F84A, el primer 1 kilobyte x 14 (0000h-03FFh) físicamente es implementado. El tener acceso a una posición anterior de la dirección físicamente puesta en práctica causará un recirculado. Por ejemplo, para posiciones 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, y 1C20h, la instrucción será la misma. El vector RESET (REPUESTO) está en la dirección 0000h y el vector INTERRUPT está en la 0004h, con cuatro fuentes de interrupción,

Hay dos bloques de memoria en el PIC16F84A, estos son la memoria de programa y la memoria de datos. Cada bloque tiene su propio bus, de modo que el acceso a cada bloque pueda ocurrir durante el mismo ciclo de oscilador. La memoria de datos adicional mayor puede ser direccionada en la RAM de objetivo general y los Registros de Función Especiales (SFRs). La operación de los SFRs que controla el “corazón” (reloj).



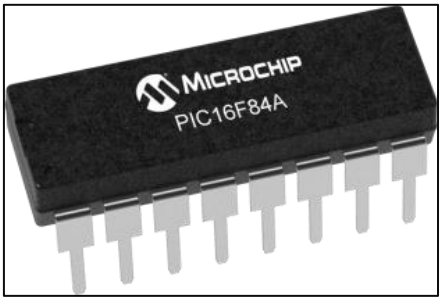
INSTRUCCIONES DEL MPLAB PARA EL PIC16F84A

Hay solo 35 instrucciones en el PIC16F84A, con códigos de instrucción de 14 bits de ancho. Todas las instrucciones ocupan una palabra y todas consumen un ciclo, excepto las de salto o bifurcación que usan dos. La velocidad máxima de funcionamiento 20MHz (200 ns x instrucción). Típicamente a 4MHz (1us x instrucción), con 1024 palabras (14 bits) de memoria de programa FLASH.

- **ADDLW** Esto significa: Agregar (sumar) el Literal al registro W (acumulador o registro de trabajo) resultado en W.
- **ADDWF** Esto significa: Suma aritmética de W y un archivo (f).
- **BCF** Esto significa: Bit Clear File (pon a “0” o aclara el bit indicado (detrás de la coma) en el archivo **f**). Ver también**BSF**.
- **BSF** Esto significa: Bit Set File (poner a 1 el bit indicado, en el archivo **f**). Ver también **BCF**.
- **BTFSC** Esto significa: Bit Test, Skip if Clear (Bit de Test, Salta si es “0”).
- **BTFSS** Esto significa: Bit Test, Skip if Set (Bit de Test, Salta si es “1”).
- **CLRF** Esto significa: Clear f [Limpia **f**] (poner a 0 los 8 bits del archivo **f**)
- **CLRW** Esto significa: Clear **W** (limpiar el registro de trabajo – llamado acumulador en otros micros)
- **CLRW** El registro W (de trabajo) es aclarado, todos los bits son puestos a 0. Cuando el **W** es puesto a 0, la bandera cero (flag **Z**) es puesta a 1, en otras palabras la bandera **Z** del registro **STATUS** es puesta a 1.
- **DECF** Esto significa: Decremento del archivo f .
- **GOTO** Esto significa: Bifurcación Incondicional.
- **INCF** Esto significa: Incrementar el archivo **f**.
- **MOVFW** Esta forma de instrucción, no es válida (**no se recomienda su uso**), a pesar de que el propio **MPLAB** la admita, significa mover el contenido del archivo **F**, al registro de trabajo **W**.
- **RETURN** Esto significa: Retorno de Subrutina. Esta instrucción está al final de una rutina o subrutina. No afecta al registro STATUS.
- **MOVLW** Esto significa: Mueve Literal a W.
- **MOVWF** Esto significa: Copia **W** al archivo llamado **f**.

MATERIALES

- PIC16F84A

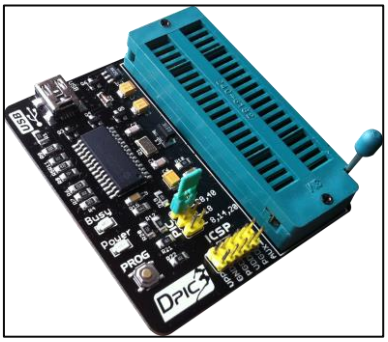


- PROTO BOARD

- LED´S

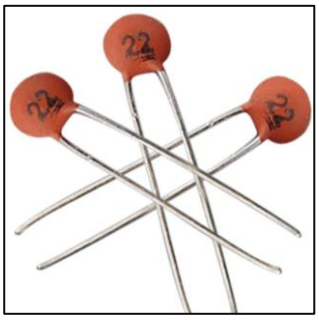
- RESISTENCIAS 110 A 330Ω

- GRABADOR DE PIC´S



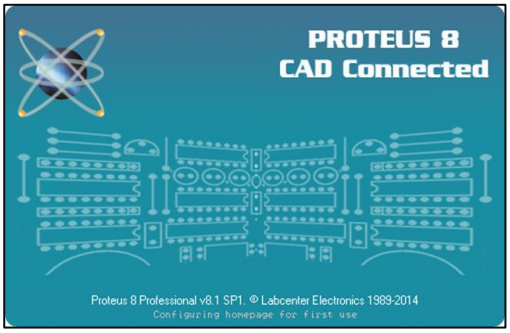
- CRISTAL DE 4 MHZ

- CAPACITORES DE 22 MHZ



- MPLAB

- PROTEUS



OBJETIVO

Obtener conocimiento sobre el lenguaje ensamblador del PIC (16F84A) es posible su utilización como base para la comprensión posterior del lenguaje de PIC's, ocupando una estructura adecuada para el desarrollo del programa.

DESARROLLO

Por el pin 2 del microcontrolador se recibe la señal analógica de un motor realice la configuración necesaria para su lectura.

CODIGO

1. Crear un nuevo archivo con extensión .ASM y nombre cualquiera
2. Crear un Proyecto nuevo eligiendo un nombre y ubicación
3. Agregar el archivo .ASM como un SOURCE FILE
4. Elegir el microcontrolador a utilizar desde SELECT DEVICE del menú CONFIGURE

C:\...\HOLAMUNDO.asm

```
__CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC
LIST P=16F84A
#INCLUDE <P16F84A.INC>

ORG 0

INICIO
BSF STATUS,RP0
MOVLW B'00000000' ;0 ENTRADAS/1 SALIDAS
MOVWF TRISA
BCF STATUS,RP0

PRINCIPAL
MOVLW B'00000010'; 0 BAJO/ CERO, LOW, 1 ALTO, HIGH,5 VOLTS
MOVWF PORTA; MUEVO EL VALOR QUE ESTA EN W Y LO PASO A PUERTO

END
```

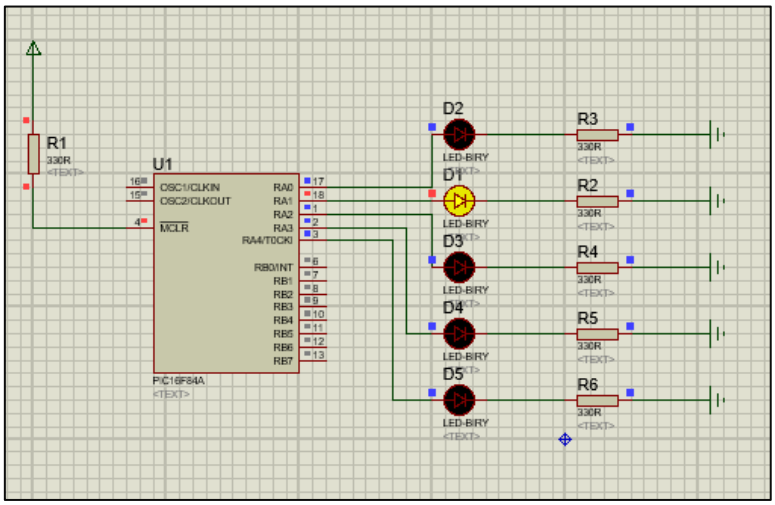
Una vez escrito y depurado el programa, se procede a la compilación. Para esto, desde el menú PROJECT se elige la opción BUILD ALL (construir todo) que, si no existen errores, devolverá un mensaje como BUILD SUCCESFULL.

Build | Version Control | Find in Files

```
Clean: Deleting intermediary and output files.
Clean: Deleted file "E:\UPIITA\MINISUMO\sumo\gio\hola.ESYM".
Clean Warning: File "E:\UPIITA\MINISUMO\sumo\gio\hola.o" doesn't exist.
Clean: Deleted file "E:\UPIITA\MINISUMO\sumo\gio\hola.ERR".
Clean: Done.
Executing: "C:\Program files\Picc\CCSC.exe" +FM "hola.c" #__DEBUG=1 +ICD +DF +LN +T +A +M
>>> Warning 202 "hola.c" Line 19(6,12): Variable never used:  nombre
>>> Warning 202 "hola.c" Line 20(6,11): Variable never used:  carga
>>> Warning 202 "hola.c" Line 21(6,12): Variable never used:  saludo
>>> Warning 202 "hola.c" Line 22(6,12): Variable never used:  valor2
>>> Warning 202 "hola.c" Line 24(7,16): Variable never used:  VoltoTemp
>>> Warning 202 "hola.c" Line 25(7,11): Variable never used:  Temp
    Memory usage:  ROM=1%    RAM=6% - 7%
    0 Errors,  6 Warnings.
Loaded E:\UPIITA\MINISUMO\sumo\gio\hola.cof.
BUILD SUCCEEDED: Mon Jun 30 18:25:16 2014
```

SIMULACION PROTEUS

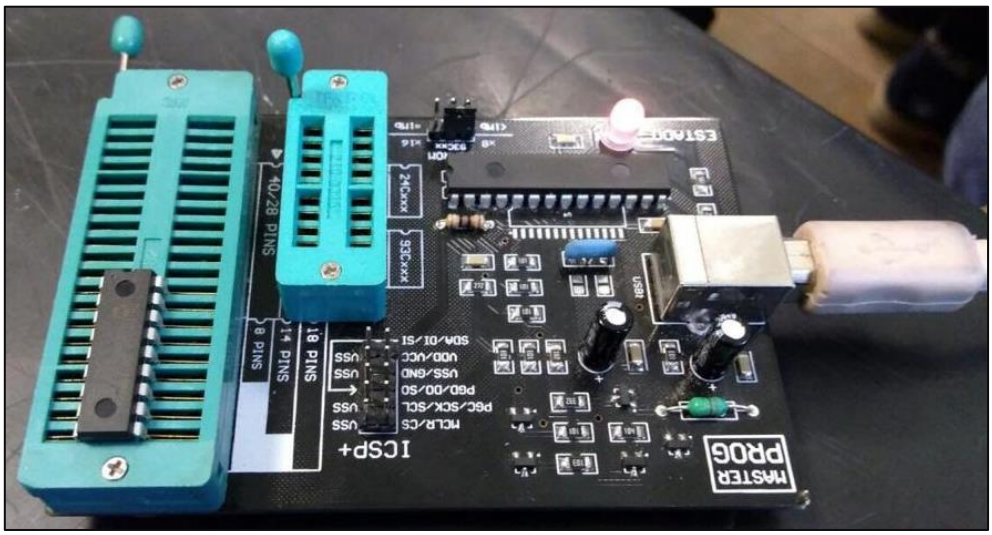
Terminada la compilación el MPLAB® nos genera un archivo de extensión .hex el cual es completamente entendible para el PIC. Es decir, solo resta grabarlo al PIC por medio de una interfaz como por ejemplo el programador Proteus. Una vez completado esto, se alimenta al mismo y el programa ya se estará ejecutando.



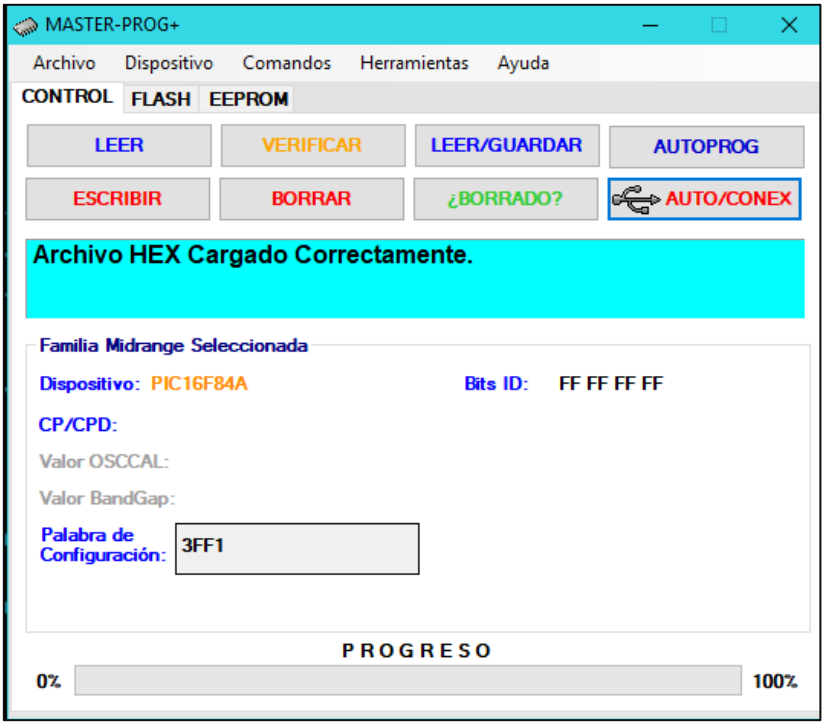
GRABAR PIC 16F84A

El programa es cargado en el microcontrolador PIC16F84A, con el grabador de pic’s después de ser probado en proteus.

1. Insertamos el pic en el grabador de pic’s y conectamos a la computadora.



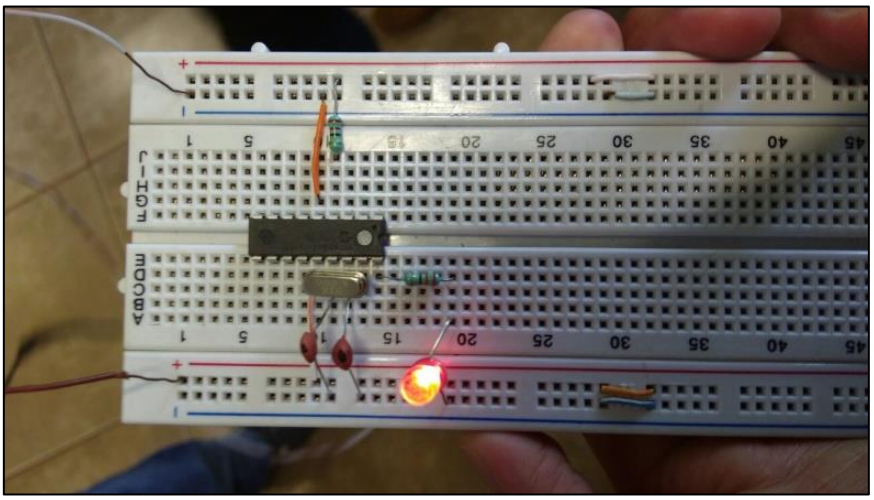
2. Cuando ya ha leído el pic cargamos el programa primero seleccionamos borrar para verificar que no contenga nada después seleccionamos escribir y cargamos el programa.



3. Esperamos y nos muestra que el programa ha sido cargo correctamente.

FISICO

Aquí conectamos todos los elementos en el protoboar en cada entrada del pic el cristal que va conectado a al pin 15 y 16 donde salen 2 capacitores que van a tierra, el pin 4 conectamos una resistencia que va a corriente, el pin 5 está conectado a la tierra, la salida es en el pin 18 que lleva una resistencia donde va conectado el positivo del led.



CONCLUSION

En esta práctica se aprendió que en base de un código llamado (lenguaje ensamblador) grabándolo en un (pic 16f84a) se puede encender un led realizado el cual fue echo en un programa llamado (MPLAB), posteriormente se realizó la simulación en el programa (PROTEUS), el cual podemos darnos cuenta como funcionara antes de poder realizarse en físico.